REMARKS

Applicants gratefully acknowledge the Examiner for courtesies extended during a telephone interview dated July 24, 2008, including co-inventor Dr. Gustavson. During this interview, Dr. Gustavson explained that the present invention involves machines having an "n-cycle penalty at the L1-L0 interface" on newer machines and is an alternative to one of the other mechanisms described in the six co-pending applications. As explained during this interview, as well as in previous interviews, the present invention can be used in conjunction with others of the seven co-pending applications listed at the beginning of the disclosure, including the concept of streaming data through the cache system in a specific manner that includes stride one memory retrieval but without the necessity for the conventional correction of the data for any of the operands. That is, the use of six kernels, rather than the conventional use of a single kernel, permits the processing of matrix data stored in the standard row or column major format without having to correct the data during retrieval for any of the operands in a level 3 multiplication.

Claims 1-19 and 21 are all the claims presently pending in the application. Claim 20 has been canceled by reason of its incorporation into the independent claims, along with clarification of the significance of the "n-cycle loading penalty" mentioned in the specification. In an attempt to expedite prosecution, new claim 21 has been added to reflect the description in lines 19-20 of page 14 of the specification related to the use of stride one without need to perform data copy, as necessitated by the conventional method of storing data in standard row-major or column-major format for at least one of the operands. As described in the specification, correction of received data results from the ability to preload data in view of the n-cycle loading penalty, and can be further enhanced by the availability of six kernels rather than the conventional usage of a single kernel.

It is noted that Applicants specifically state that no amendment to any claim herein should be construed as a disclaimer of any interest in or right to an equivalent of any element or feature of the amended claim.

Claim 20 stands rejected under 35 U.S.C. § 112, second paragraph, as allegedly indefinite. Applicants have amended this claim language, as incorporated into the independent claims, to address this issue and respectfully request that the Examiner reconsider and withdraw this rejection.

Claims 1, 2, (17), and 20 stand rejected under 35 U.S.C. § 103(a) as unpatentable over

U.S. Patent No. 5,438,669 to Nakazawa et al., further in view of US Patent 6,115,730 to Dhablania et al., and further yet in view of newly-cited US Patent 6,507,892 to Mulla et al. Claims 3-16, 18, and 19 stand rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over Nakazawa/Dhablania, further in view of Dongarra, et al., "A Set of Level 3 Basic Linear Algebra Subprograms" (and presumably, further in view of newly-cited Mulla).

These rejections are respectfully traversed in the following discussion, since, as discussed during the above-mention telephone interview, the present invention includes aspects unique to newer architectures, including an FPU having an "*n*-cycle loading penalty" at the L0/L1 interface.

I. THE CLAIMED INVENTION

The claimed invention is directed to a <u>software</u> method of <u>improving at least one of efficiency and speed</u> in executing a linear algebra subroutines <u>on a computer having a floating point unit (FPU) capable of overlapping loading data and processing of the data</u>. A load instructions are used to preload data in a timely manner into floating point registers (FRegs) of the FPU <u>from the L1 cache</u> from an execution code controlling operation of the FPU that is performing the linear algebra subroutines in a near optimal way.

In a programming language, this can be done by "unrolling", which preloads data in advance into the FRegs and then executes corresponding FReg instructions of the linear algebra subroutine.

As explained beginning at line 17 of page 3 of the specification, a number of advancements have been made in the computerized processing of linear algebra routines on computers of various architectures.

The claimed invention addresses one of the problems identified by the present inventors, given a specific interface with a recent floating point unit (FPUs) design that typically is used for such linear algebra routine processing, and is one of several alternate solutions to this problem that is described in the seven co-pending applications.

II. THE PRIOR ART REJECTIONS

The Examiner alleges that Nakazawa, when modified by Dhablania and further yet modified by newly-cited Mulla, renders obvious the present invention described by claims 1, 2, (17), and 20, and, when further modified by Dongarra, renders obvious claims 3-16, 18, and 19.

Applicants again submit, however, that there are elements of the claimed invention which are neither taught nor suggested by Nakazawa and that the urged combination of this reference with Dhablania and/or Mulla would be improper hindsight, given the information and machine configuration described in primary reference Nakazawa, and would fail to satisfy the plain meaning of the language of the claims.

As explained in the second sentence (e.g., in column 1 at lines 12-17), Nakazawa addresses a processing method in a computer architecture in which <u>cache is not considered so effective</u>. In this architecture, a large number of data registers accessible to main memory are used. To overcome this design deficiency, Nakazawa introduces preloading of data from main memory directly into these data registers.

In contrast, in the present invention, the L1 cache <u>is</u> used to receive the matrix data transfer from main memory and the FPUs. Then, the data is fast loaded into the FRegs and rearranged in the FRegs to be correct in less than n cycles by using instructions available in the newer architectures. Therefore, the problem being addressed by the solution of the present invention <u>includes an n-cycle penalty at the L0/L1 interface</u>. As explained below, this n-cycle penalty is due to the constraint that standard data format is conventionally used for storing matrix data, so that matrix data arrives into the FPU in an order that is not correct for the processing. Whether the reason that Nakazawa considers that "a cache is not so effective" is related in any way to the n-cycle penalty of the architecture of the present invention is not clear from the description therein, but Applicants submit that, in fact, this problem did not even exist for Nakazawa.

However, it is very clear that, with its intent to <u>completely eliminate its cache during matrix processing</u>, Nakazawa <u>clearly teaches against</u> the approach of the present invention and clearly offers another solution to receiving data into its FPUs, since the present invention addresses its L0/L1 problem by a preloading concept <u>rather than bypassing the cache</u>. That is, the present invention clearly addresses the data preloading into the FPUs <u>using the cache</u> and does so using a software mechanism that does not require the additional hardware registers of the Nakazawa computer architecture. As pointed out below, another method of solving the problem of data being out-of-order is to rearrange the data in the memory to be stored in a nonstandard format that is predetermined to have the data in correct order upon its arrival into the FPU, taking into account that the computer language will transport the data in its conventional manner.

At the top of page 4 of the latest Office Action, the Examiner alleges "*It is obvious*

that said method/idea can be implemented/realized using either hardware solution or using software solution."

In response, Applicants respectfully disagree with the above-recited Examiner's conclusory statement that implies that implementation using software would inherently be obvious as a substitute for a hardware mechanism. That is, even if one is able to achieve a similar effect (e.g., prefetching or preloading) using either hardware or software, it is clear that the principle of operation would inherently differ in these two approaches, so that similar effect does not in any way imply obviousness, as the Examiner seems to consider.

The Examiner's conclusory statement improperly attempts to redefine obviousness as based upon whether a similar result is achieved, which is not the correct standard for determining obviousness. Rather, the Examiner's initial burden in the present evaluation is to articulate a reasonable rationale to modify the primary reference, e.g., Nakazawa, to arrive at the claimed invention, as clearly confirmed in the recent US Supreme Court holding in *KSR*.

Perhaps most important in this consideration is that the hardware mechanism of primary reference Nakazawa intentionally attempts to bypass cache, since its fundamental solution is to add registers to the FPU so that data can be pre-loaded directly from memory into "windows" of registers serving different loops of the processing, as described in the Abstract. There clearly is no suggestion in Nakazawa of rearranging the data in the coprocessor, using up to n cycles.

The approach in Nakazawa is clearly fundamentally different from the claimed invention. Indeed, the approach in Nakazawa can only be considered as demonstrating that there is more than one solution to the more general problem recognized by the present inventors that having matrix data stored in memory in the conventional standard format causes inefficiencies for processing that data, and, as demonstrated in the seven co-pending applications, there are several ways to address this problem, some of which would work together synergistically.

First, although unrolling of instructions is used in both Nakazawa and the claimed invention, the claimed invention requires that the cache system be used for the matrix data transfer. In the terminology of the seven co-pending applications, and as explained to the Examiner during previous telephone interviews, the preloading of data from memory into L1 cache is referred to in these co-pending applications as "pre-fetching" or "streaming", whereas the preloading of data from the L1 cache into the FRegs of the FPU is referred to as "preloading", which is a primary aspect of the claimed invention under evaluation. This

preloading from L1 into the FRegs (which the co-inventors refer to as the "L0" cache) is one of the alternatives developed by the present inventors to overcome the n-cycle penalty at the L0/L1 interface and constitutes a software solution for this problem, rather than forcing a redesign of the hardware. As explained below, an alternative approach would be to initially rearrange the data in memory to be in a nonstandard data structure, which the inventors describe in a co-pending application as the "register block" format, since it is predetermined to be the format appropriate for loading data as desired into the FPUs with no correction or rearrangement needed. Thus, the present invention is reasonably described as being one of two nonobvious alternatives to the method used in Nakazawa.

To better understand the significance of the present invention and the distinction from the cited prior art, Applicants explain that in the modern computer architecture toward which the present invention is directed:

(1) Data has to arrive in a timely manner in L0 (e.g., the FRegs) for use by the vector FMA units.

(2) There is a limited (called stride 1) way to do fast (vector) loads from L1 to L0.

(3) For pre-loading, data has arrived in L1 in an unsatisfactory manner to do fast (vector) loads to L0. Here is what pre-loading does:

a) It does fast (vector) loads to L0, implying that use by the vector FMA units will produce incorrect results.

b) Then, the "n-cycle penalty" is executed, as follows:

c) The L0 unit in these architectures has available move operations (e.g., as described in the specification, "load" operations) that can rearrange data amongst its registers.

d) These move operations are used to rearrange the fast (vector) loads of a) so that now use of the vector FMA units will produce correct results.

Note: the move operations within L0 require time. This required time is the "n-cycle penalty" mentioned in the specification and added into the independent claims, along with a clarification of this terminology.

The seven co-pending applications include two ways to correct data to L0:

i) preloading, which is described above and is the object of the present application and the claimed invention, and

ii) prefetching, in which one prepares the data ahead of time in memory using

the new format referred to the present inventors as the "register block format", and is the object of co-pending Application S/N 10/671,888. In this alternative approach, data is then streamed to L1 in a correct format for fast loading into L0 in the correct format for the fast vector FMA units.

Therefore, in summary, the first method i) puts data in the correct format in L0, which is the target; and the second method ii) puts data in the correct format in memory (via register blocking), which is the source. In both cases the data has been formatted wrong by the compiler using the standard API. And, in both cases, the object is to overcome the problem of having to use standard data format for storing matrix data.

Returning now to the rejection of record, since primary reference Nakazawa expressly bypasses its cache system, there clearly is no *n*-cycle loading penalty at its L0/L1 interface during the described processing, and thereby it is clear that the claimed invention is non-obvious over Nakazawa.

Moreover, to even modify primary reference Nakazawa to satisfy the plain meaning of the claim language of even the independent claims would clearly change the principle of operation of Nakazawa and would, therefore, be improper. Therefore, secondary reference Dhablania and tertiary reference Mulla would not overcome the fundamental deficiencies of primary reference Nakazawa.

Moreover, the preloading instructions of Nakazawa retrieve individual words from memory directly into a register of the appropriate window being preloaded, as is clearly described in lines 61-65 of column 14: "*3. Floating point register preload instruction: (Instruction mnemonic) FLDPRM a (GRm), FRn (Function) Reads 8-byte data from the main memory address specified by the value of the general register in and stores it in the floating point register n.*"

Finally, as described in new claim 21, the claimed invention is actually only one aspect of a larger scheme of retrieving data from memory more efficiently than using individual word retrievals, such as demonstrated by the above-described preloading instruction. That is, as a preliminary to the preloading of data from L1 cache into L0, the data of the claimed invention is retrieved in the much more efficient stride one retrieval, described in lines 14-17 of page 12 of the specification as referring the line-size increment of data retrieval of a format having data contiguously arranged so as to honor double-word boundaries.

The efficiency of the stride one retrieval is even further enhanced by selectively using

six kernels, rather than the conventional single kernel, for processing. This use of alternate kernels eliminates the additional data copying processing necessary for one or more operands due to the standard row/column major format required to be used in conventional higher level computer languages, as described at lines 7-23 of page 14 of the specification.

Therefore, Applicants respectfully submit that the claimed invention includes a number of features that are clearly not present in Nakazawa and that cannot be incorporated into the method of Nakazawa without fundamentally changing its principle of operation and/or defeating its intended and express purpose of eliminating its cache interface during matrix processing by its FPU.

In summary, the method in Nakazawa provides a hardware solution to get matrix data from memory to the processor and will work only for the 1995 hardware described therein. In contrast, the present invention provides a general software solution for matrix multiplication, possibly in combination with one or more of the six other co-pending applications.

Therefore, Nakazawa teaches clearly that matrix data using his processing unit cannot be efficiently retrieved from memory using typical cache architecture. In contrast, the present invention provides basic ground rules for preloading in the context of matrix multiplication kernels. This is novel and not obvious from Nakazawa. That is, relative to independent claim 1 (as well as remaining independent claims), Nakazawa would work only on the now-obsolete 1995 hardware, whereas the present invention is much faster and works with standard cache-based machines.

Hence, turning to the clear language of the claims, in Nakazawa there is no teaching or suggestion of: ") A <u>software</u> method of improving at least one of efficiency and speed in executing a linear algebra subroutine on a computer having a floating point unit (FPU) with floating point registers (FRegs) and a load/store unit (LSU) capable of overlapping loading data and processing said data by the FPU, said FPU being interfaced with an L1 cache and <u>having an L1 cache/FReg interface "loading penalty of *n* cycles</u>", *n* being an integer greater than or equal to 1, <u>during which data is rearranged in up to *n* cycles in said FRegs because data is out of order for said processing</u>, said method comprising: loading matrix data from a memory through a cache system at a <u>fastest possible rate</u>; and for an execution code controlling operation of said linear algebra subroutine execution, <u>overlapping by preloading data into said FRegs of said FPU and then rearranging the data in the FRegs for up to said *n* cycles</u>, said overlapping causing said matrix data to arrive into said FRegs from said L1 cache

to be timely executed by the FPU operations of said linear algebra subroutine on said FPU", as required by independent claim 1. The remaining independent claims have similar language.

Therefore, Applicant submits that there are elements of the claimed invention that are not taught or suggested by Nakazawa. Therefore, the Examiner is respectfully requested to withdraw this rejection for claims 1, 2, 17, and 20.

Claims 3-16, 18, and 19 stand rejected as unpatentable over Nakazawa, further in view of Dongarra. However, regardless of the propriety of combining Dongarra with Nakazawa, this secondary reference does not overcome the basic deficiency identified above for Nakazawa, so that these claims are also clearly patentable over Nakazawa.

Moreover, the following comments relate to the non-obviousness of additional dependent claims over the prior art of record.

Relative to claim 4, as discussed on page 10 of the latest Office Action, Applicants again respectfully point out that use of L1 BLAS and what followed, L2 BLAS, do not work efficiently on today's cache-based architecture. The reason is that they are very slow methods. Nakawaza and Dhablania do not disclose the method of Claim 1. Also, to paraphrase Dongarra, when he asks suppliers to "build a better mouse trap", the present invention <u>has</u> built that better mouse trap.

Relative to claim 5, Applicants again point out that Nakazawa and, particularly, Dongarra do not use level L3 routines for factorization *per se*, and their methods are very slow compared to those used in the present invention and the six other co-pending applications. It was not obvious to Dongarra to use fast methods for factorization *per se*.

Relative to claim 18, Applicants again point out that Dongarra does <u>not</u> disclose any detail on <u>how to implement</u> fast BLAS. Dongarra is actually <u>asking</u> computer manufacturers to implement his methods, rather than disclosing fast methods to actually do this.

Finally, relative to secondary reference Dongarra, Applicants again point out that they are not attempting to patent matrix multiplication *per se*. Level 3 BLAS are an industry standard for matrix multiplication. Cayley defined matrix multiplication in 1854 for the first time. Dongarra merely repeats that definition in a very slow implementation of matrix multiplication.

Therefore, Applicants submit that all claims are clearly patentable over Nakazawa, even if combined by Dongarra and newly-cited Mulla, and respectfully request the Examiner to reconsider and withdraw these rejections.

III. FORMAL MATTERS AND CONCLUSION

In view of the foregoing, Applicant submits that claims 1-19 and 21, all the claims presently pending in the application, are patentably distinct over the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue at the earliest possible time.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed below to discuss any other changes deemed necessary in a telephonic or personal interview.

The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Respectfully Submitted,

Date: October 10, 2008

Frederick E. Cooperrider
Registration No. 36,769

McGinn Intellectual Property Law Group, PLLC
8321 Old Courthouse Road, Suite 200
Vienna, VA 22182-3817
(703) 761-4100
Customer No. 21254